PIO-D144/D168

User's Manual

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, not for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 2003 by ICP DAS. All rights are reserved.

Trademark

The names used for identification only may be registered trademarks of their respective companies.

Tables of Contents

1. Introduction 4

1.1 Specifications 4
1.2 Features 5
1.3 Product Check List 6

2. Hardware configuration 7

2.1 Board Layout 7
2.2 I/O Port Location 8
2.3 Pin Assignments 9
2.4 Enable I/O Operation 11
2.5 D/I/O Architecture 12
2.6 Interrupt Operation 13
2.7 Daughter Boards 14
2.7.1 DB-37 14
2.7.2 DN-37 & DN-50 14
2.7.3 DB-8125 15
2.7.4 ADP-37/PCI & ADP-50/PCI 15
2.7.5 DB-24P, DB-24PD Isolated Input Board 16
2.7.6 DB-24R, DB-24RD Relay Board 17
2.7.7 DB-24PR, DB-24POR, DB-24C 18
2.7.8 Daughter Board Comparison Table 19

3. I/O Control Register 20

3.1 How to Find the I/O Address 20
3.2 The Assignment of I/O Address 23
3.3 The I/O Address Map 24
3.3.1 RESET\ Control Register 25
3.3.2 AUX Control Register 25
3.3.3 AUX data Register 25
3.3.4 INT Mask Control Register 26
3.3.5 Aux Status Register 26
3.3.6 Interrupt Polarity Control Register 26
3.3.7 Read/Write 8-bit data Register 27
3.3.8 Active I/O Port Control Register 27
3.3.9 I/O Selection Control Register 28

4. Software Installation 30

4.1 Software Installing Procedure 30
4.2 PnP Driver Installation 30

5. DLL Function Description 31

5.1 Table of ErrorCode and ErrorString 32
5.2 Function Descriptions 32
5.3 FUNCTIONS OF TEST 33
5.3.1 PIODIO_GetDllVersion 33

5.3.2 PIODIO_ShortSub 34
5.3.3 PIODIO_FloatSub 34
5.4 Digital I/O FUNCTIONS 35
5.4.1 PIODIO_OutputByte 35
5.4.2 PIODIO_InputByte 35
5.4.3 PIODIO_OutputWord 36
5.4.4 PIODIO_InputWord 36
5.5 Driver Relative Functions 37
5.5.1 PIODIO_GetDriverVersion 37
5.5.2 PIODIO_DriverInit 37
5.5.3 PIODIO_SearchCard 38
5.5.4 PIODIO_DriverClose 38
5.5.5 PIODIO_GetConfigAddressSpace 39
5.6 INTERRUPT FUNCTION 40
5.6.1 PIODIO_IntResetCount 40
5.6.2 PIODIO_IntGetCount 40
5.6.3 PIODIO_IntInstall 41
5.6.4 PIODIO_IntRemove 42
5.6.5 Architecture of Interrupt mode 42
5.6.6 Program Architecture 43

6. Demo Programs for Windows 44
6.1 Digital Output of CN1(Port0~Port2) 45
6.2 Digital input /output of CN5 and CN6 45
6.3 Digital output / input of all Ports (Port0~20) 46
6.4 Interrupt of P2C0 46

Appendix 47
Appendix A. Related DOS Software 47
A-1 Where is the related software 47
A-2 DOS LIB Function 48

1. Introduction

The PIO-D144/D168 consists of one DB-37connector and five 50-pin flat-cable connectors/six 50-pin flat-cable connectors. Note that there are three 8-bit ports, PA, PB & PC in each connector. Every port can be programmable and configured as 8-bit Digital input or output at the same time. Therefore, the PIO-D144/D168 can provide 144/168 channels of TTL-compatible D/I/O.

1.1 Specifications

Model Name	PIO-D144
Programmable Digital I/O	
Channels	144
Digital Input	
Compatibility	5 V/TTL
Input Voltage	Logic 0: 0.8 V max. Logic 1: 2.0 V min.
Response Speed	1.2 MHz (Typical)
Digital Output	
Compatibility	5 V/TTL
Output Voltage	Logic 0: 0.4 V max. Logic 1: 2.4 V min.
Output Capability	Sink: 0.8 mA @ 0.8 V Source: -2.4 mA @ 2.0 V
Response Speed	1.2 MHz (Typical)
General	
Bus Type	5 V PCI, 32-bit, 33 MHz
Data Bus	8-bit
Card ID	No
I/O Connector	Female DB37 x 1 50-pin box header x 5
Dimensions (L x W x D)	180 mm x 105 mm x22 mm
Power Consumption	1100 mA @ +5 V
Operating Temperature	0 ~ 60 °C
Storage Temperature	-20 ~ 70 °C
Humidity	5 ~ 85% RH, non-condensing

Model Name	PIO-D168
Programmable Digital I/O	
Channels	168
Digital Input	
Compatibility	5 V/TTL
Input Voltage	Logic 0: 0.8 V max. Logic 1: 2.0 V min.
Response Speed	1.2 MHz (Typical)
Digital Output	
Compatibility	5 V/TTL
Output Voltage	Logic 0: 0.4 V max. Logic 1: 2.4 V min.
Output Capability	Sink: 0.8 mA @ 0.8 V Source: -2.4 mA @ 2.0 V
Response Speed	1.2 MHz (Typical)
General	
Bus Type	5 V PCI, 32-bit, 33 MHz
Data Bus	8-bit
Card ID	No
I/O Connector	Female DB37 x 1 50-pin box header x 6
Dimensions (L x W x D)	200 mm x 105 mm x 22 mm
Power Consumption	1300 mA @ +5 V
Operating Temperature	0 ~ 60 °C
Storage Temperature	-20 ~ 70 °C
Humidity	5 ~ 85% RH, non-condensing

1.2 Features

- PC compatible PCI bus.
- PIO-D144 : One DB-37 connector and five 50-pin flat-cable connectors.
  PIO-D168 : One DB-37 connector and six 50-pin flat-cable connectors.
- Each port consists of three 8-bit ports, PA, PB & PC in every connector.
- Each port can be independently configured as DI or DO at the same time.
- PIO-D144 board : 6 connectors = 6×3 ports = 6×3×8 bits =144 bits.
  PIO-D168 board : 7 connectors = 7×3 ports = 7×3×8 bits =168 bits.
- 4 interrupt sources: P2C0, P2C1, P2C2, P2C3.

1.3 Product Check List

Your package includes the following items:

- One PIO-D144/D168 card
- One company floppy diskette or CD.
- One Quick Start Guide

It is recommended to read the Quick Start Guide first. All the necessary and essential information are given in the Quick Start Guide as follows:

- Where to get the software driver, demo programs and other resources.
- How to install the software.
- How to test the card.

Attention !

If any of these items is missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton in case you want to ship or store the product in the future.

2. Hardware configuration

2.1 Board Layout

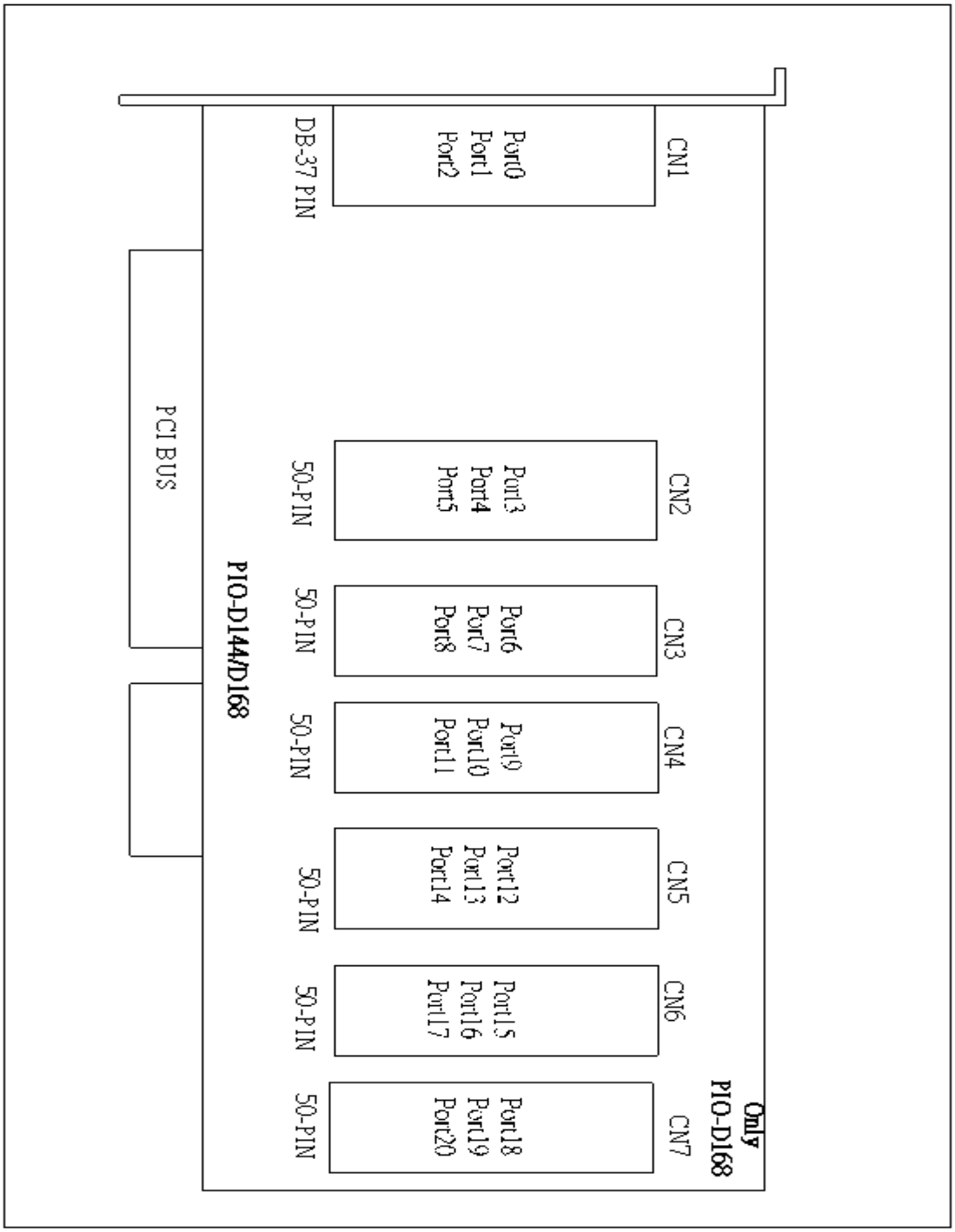

Figure 2.1

2.2 I/O Port Location

There are 18/21 8-bit I/O ports in the PIO-D144/D168. Every port can be independently configured as D/I or D/O port. When the PC is first power-on, all ports are set as Digital input port. Therefore, user needs to configure these ports as digital input or output port before application. The I/O port is named as following table and its location can be found in Figure 2.1.

Table 2.1

Connector of PIO-D144/D168	PA0 ~ PA7	PB0 ~ PB7	PC0 ~ PC7
CN1	Port0	Port1	Port2
CN2	Port3	Port4	Port5
CN3	Port6	Port7	Port8
CN4	Port9	Port10	Port11
CN5	Port12	Port13	Port14
CN6	Port15	Port16	Port17
CN7 (PIO-D168 Only)	Port18	Port19	Port20

Refer to Sec. 2.1 for board layout & I/O port location.

Note: P2C0, P2C1, P2C2, P2C3 of CN1 can be used as interrupt signal source. Refer to Sec. 2.6 and 2.3 pin assignments for more information.

2.3 Pin Assignments

The Pin assignments for all connectors of PIO-D144/D168 are represented as Table 2.2 and 2.3. All signal source of each digital input or output pin (channel) are TTL compatible. **Note that CN7(Port18~Port20) is only for PIO-D168.**

Table 2.2 CN1: 37-PIN D-type female connector for Port0~Port2

Pin Number	Description	Pin Number	Description
1	N. C.	20	Vcc
2	N. C.	21	GND
3	P1B7	22	P2C7
4	P1B6	23	P2C6
5	P1B5	24	P2C5
6	P1B4	25	P2C4
7	P1B3	26	P2C3
8	P1B2	27	P2C2
9	P1B1	28	P2C1
10	P1B0	29	P2C0
11	GND	30	P0A7
12	N.C.	31	P0A6
13	GND	32	P0A5
14	N.C.	33	P0A4
15	GND	34	P0A3
16	N.C.	35	P0A2
17	GND	36	P0A1
18	VCC	37	P0A0
19	GND	XXXXXXX	This pin not available

Table 2.3 CN2/3/4/5/6/7: 50-PIN of flat-cable connector for Port3~Port20

Pin Number	Description	Pin Number	Description
1	P5C7/ P8C7/ P11C7/ P14C7/ P17C7/ P20C7	2	GND
3	P5C6/ P8C6/ P11C6/ P14C6/ P17C6/ P20C6	4	GND
5	P5C5/ P8C5/ P11C5/ P14C5/ P17C5/ P20C5	6	GND
7	P5C4/ P8C4/ P11C4/ P14C4/ P17C4/ P20C4	8	GND
9	P5C3/ P8C3/ P11C3/ P14C3/ P17C3/ P20C3	10	GND
11	P5C2/ P8C2/ P11C2/ P14C2/ P17C2/ P20C2	12	GND
13	P5C1/ P8C1/ P11C1/ P14C1/ P17C1/ P20C1	14	GND
15	P5C0/ P8C0/ P11C0/ P14C0/ P17C0/ P20C0	16	GND
17	P4B7/ P7B7/ P10B7/ P13B7/ P16B7/ P19B7	18	GND
19	P4B6/ P7B6/ P10B6/ P13B6/ P16B6/ P19B6	20	GND
21	P4B5/ P7B5/ P10B5/ P13B5/ P16B5/ P19B5	22	GND
23	P4B4/ P7B4/ P10B4/ P13B4/ P16B4/ P19B4	24	GND
25	P4B3/ P7B3/ P10B3/ P13B3/ P16B3/ P19B3	26	GND
27	P4B2/ P7B2/ P10B2/ P13B2/ P16B2/ P19B2	28	GND
29	P4B1/ P7B1/ P10B1/ P13B1/ P16B1/ P19B1	30	GND
31	P4B0/ P7B0/ P10B0/ P13B0/ P16B0/ P19B0	32	GND
33	P3A7/ P6A7/ P9A7/ P12A7/ P15A7/ P18A7	34	GND
35	P3A6/ P6A6/ P9A6/ P12A6/ P15A6/ P18A6	36	GND
37	P3A5/ P6A5/ P9A5/ P12A5/ P15A5/ P18A5	38	GND
39	P3A4/ P6A4/ P9A4/ P12A4/ P15A4/ P18A4	40	GND
41	P3A3/ P6A3/ P9A3/ P12A3/ P15A3/ P18A3	42	GND
43	P3A2/ P6A2/ P9A2/ P12A2/ P15A2/ P18A2	44	GND
45	P3A1/ P6A1/ P9A1/ P12A1/ P15A1/ P18A1	46	GND
47	P3A0/ P6A0/ P9A0/ P12A0/ P15A0/ P18A0	48	GND
49	V_{CC}	50	GND

2.4 Enable I/O Operation

When the PC is first power-on, all operations of digital I/O channels of each port are disabled. Note that the digital I/O channel of each port is enabled or disabled by the RESET\ signal, refer to Sec. 3.3.1 for more information. The power-on states for all DI/O ports are given as following:

- D/I/O operations of each port are all disabled.
- D/I/O ports are all configured as Digital input port.
- D/O latch register are all undefined, refer to Sec. 2.5.

The user has to perform some initialization before using these digital I/O ports. The recommended steps are given as following:

Step 1: Find address-mapping of PIO/PISO cards. (Refer to Sec.3.1)
Step 2: Enable all Digital I/O operation. (Refer to Sec. 3.3.1).
Step 3: Select the controlled port (Refer to Sec. 3.3.8).
Step 4: Send initial-value to the D/O latch register of this controlled port. (Refer to Sec. 2.5 & Sec. 3.3.7)
Step 5: Repeat Step3 & Step4 to initiate the other D/O ports.
Step 6: Configure all Digital I/O ports to their expected D/I or D/O function. (Refer to Sec. 3.3.9)

For more information of initial procedure for digital I/O port, please refer to DEMO1.C demo program.

2.5 D/I/O Architecture

The digital I/O control architecture for PIO-D144 /D168 is demonstrated in Figure 2.2. The operation method of control signal is presented as below.

- The RESET\ is in Low-state means that all D/I/O operation is disabled.
- The RESET\ is in High-state means that all D/I/O operation is enabled.
- If D/I/O is configured as D/I port, the port can accept digital input from external signal source.
- If D/I/O is configured as D/O port, the digital output value can be read back from the port.
- If D/I/O is configured as D/I port, sending data to Digital input port will change the D/O latch register only. And the latched data will be output when the port is configured as digital output and is activated right away.

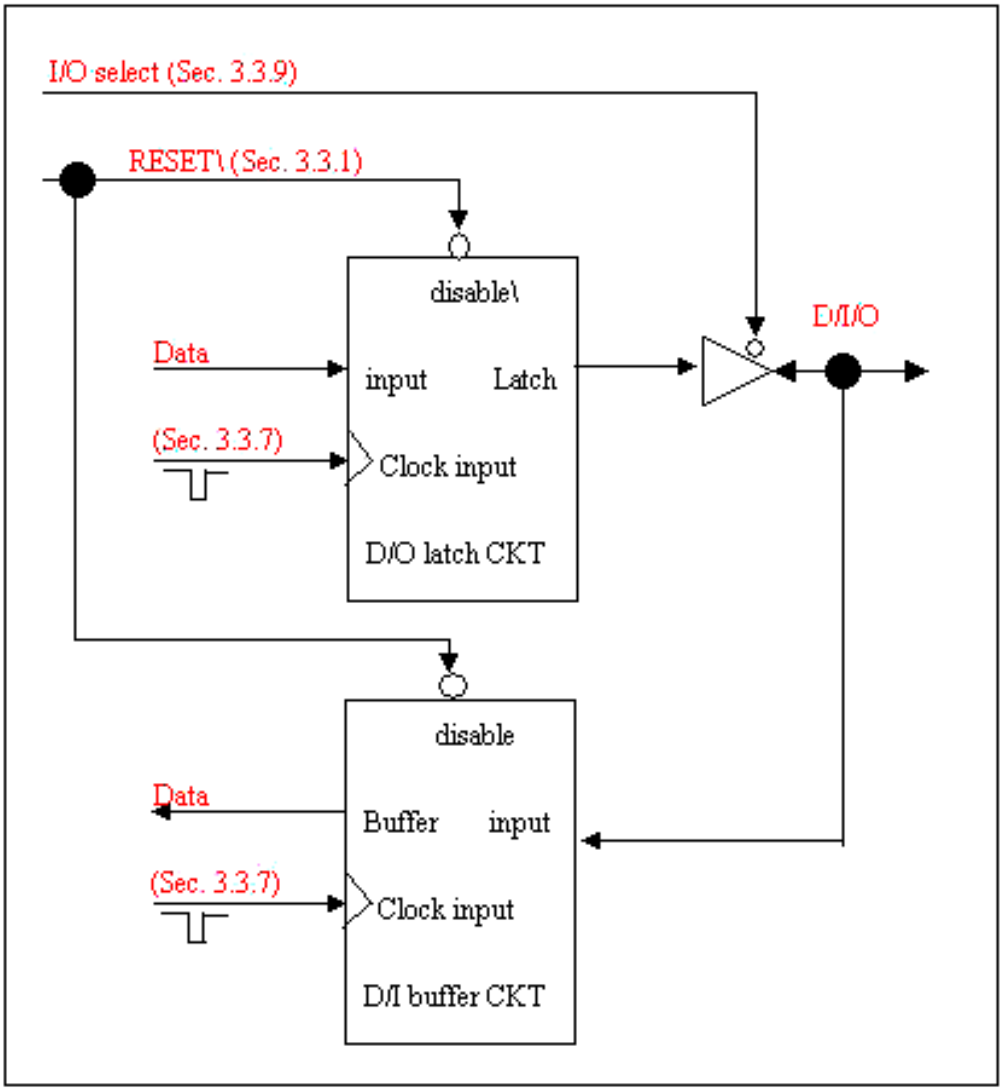

Figure 2.2

2.6 Interrupt Operation

The P2C0, P2C1, P2C2 and P2C3 of CN1 can be used as interrupt signal source. Refer to Sec. 2.1 for P2C0/P2C1/P2C2/P2C3 location. The interrupt of PIO-D144/D168 is **level-trigger and Active_High**. The interrupt signal can be programmable as **inverted or non-inverted**. The procedures for how to configure the interrupt signal source are given as follows:

1. Make sure **the initial level is High or Low from the signal source.**
2. If the initial state is High, please select the **inverted** setting for interrupt signal source (Section. 3.3.6). If the initial state is Low, please select the **non-inverted** setting for interrupt signal source (Section. 3.3.6)
3. Enable the interrupt function (Section. 3.3.4)
4. If the interrupt signal is active, the interrupt service routine will be started up.

Note that DEMO3.C & DEMO4.C are demo programs for single interrupt source and DEMO5.C is the demo program for four interrupt sources in DOS operating system. If only one interrupt signal source is used, the interrupt service routine does not need to identify the interrupt source. (Refer to DEMO3.C & DEMO4.C). However, if there are more than one interrupt source, the interrupt service routine has to identify the active signals as following: (refer to DEMO5.C)

1. Read the new status of the interrupt signal source
2. Compare the new status with the old status to identify the active signals
3. If P2C0 is active, service P2C0 & non-inverter/inverted the P2C0 signal
4. If P2C1 is active, service P2C1 & non-inverted/inverted the P2C1 signal
5. If P2C2 is active, service P2C2 & non-inverted/inverted the P2C2 signal
6. If P2C3 is active, service P2C3 & non-inverted/inverted the P2C3 signal
7. Save the new status to old status

Limitation: if the interrupt signal is too short, the new status may be as same as old status. So the interrupt signal must be hold active until the interrupt service routine is executed. This hold time is different for different O.S. The hold time can be as short as micro-second or as long as second. In general, 20 ms is enough for all O.S.

2.7 Daughter Boards

2.7.1 DB-37

The DB-37 is a general purpose daughter board for D-sub 37 pins. It is designed for easy wire connection by pin-to-pin.

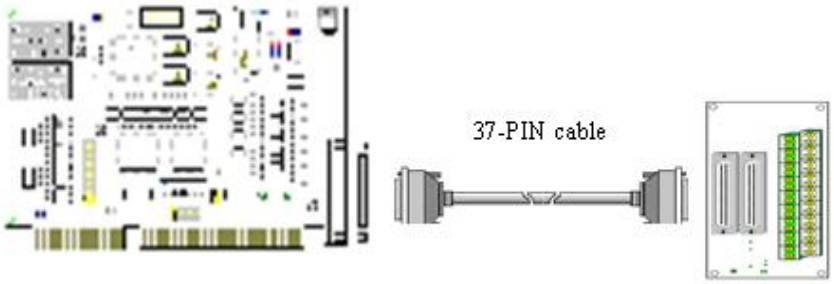

Figure 2.3

2.7.2 DN-37 & DN-50

The DN-37 is a general purpose daughter board for DB-37 with DIN-Rail Mounting. The DN-50 is designed for 50-pin flat-cable header win DIN-Rail mounting. They are also designed for easy wire connection by pin-to-pin.

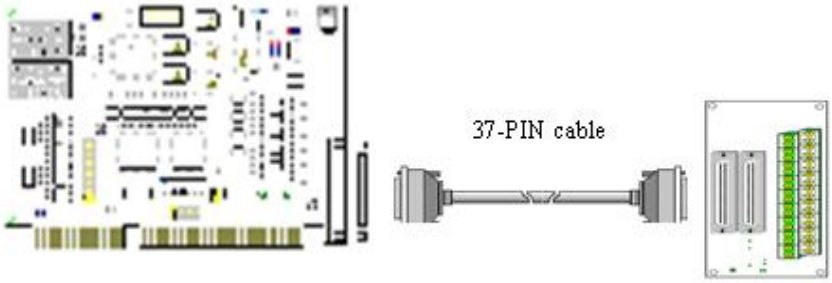

Figure 2.4

2.7.3 DB-8125

The DB-8125 is a general purpose screw terminal board. It is designed for easy wire connection. There are one DB-37 & two 20-pin flat-cable headers in the DB-8125.

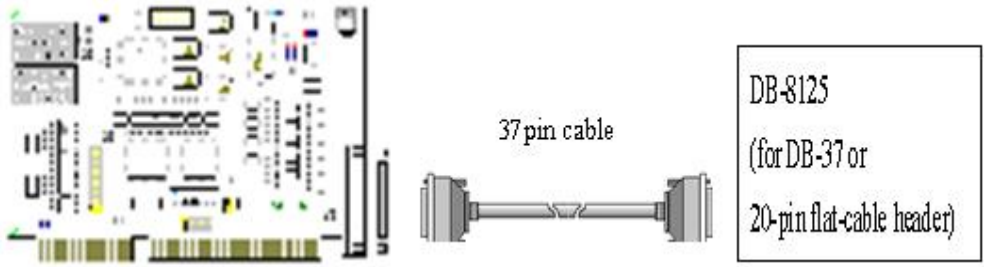

Figure 2.5

2.7.4 ADP-37/PCI & ADP-50/PCI

The ADP-37/PCI & ADP-50/PCI are extender for 50-pin header. One side of ADP-37/PCI and ADP-50/PCI can be connected to a 50-pin header. The other side can be mounted on the PC chassis as shown in the following figure. Note that ADP-37/PCI is 50-pin header to DB-37 extender and ADP-50/PCI is 50-pin header to 50-pin header extender.

Figure 2.6

2.7.5 DB-24P, DB-24PD Isolated Input Board

The DB-24P is a 24-channel isolated digital input daughter board. The optically isolated inputs of the DB-24P consist of a bi-directional optocoupler with a resistor for current sensing. You can use the DB-24P to sense DC signal from TTL levels up to 24 V or use the DB-24P to sense a wide range of AC signals. You can also use this board to isolate the computer from large common-mode voltage, ground loops and transient voltage spike that often is occurred in industrial environments, as shown in Figure 2.7. Table 2.4 is the comparison of DB-24P and DB-24PD.

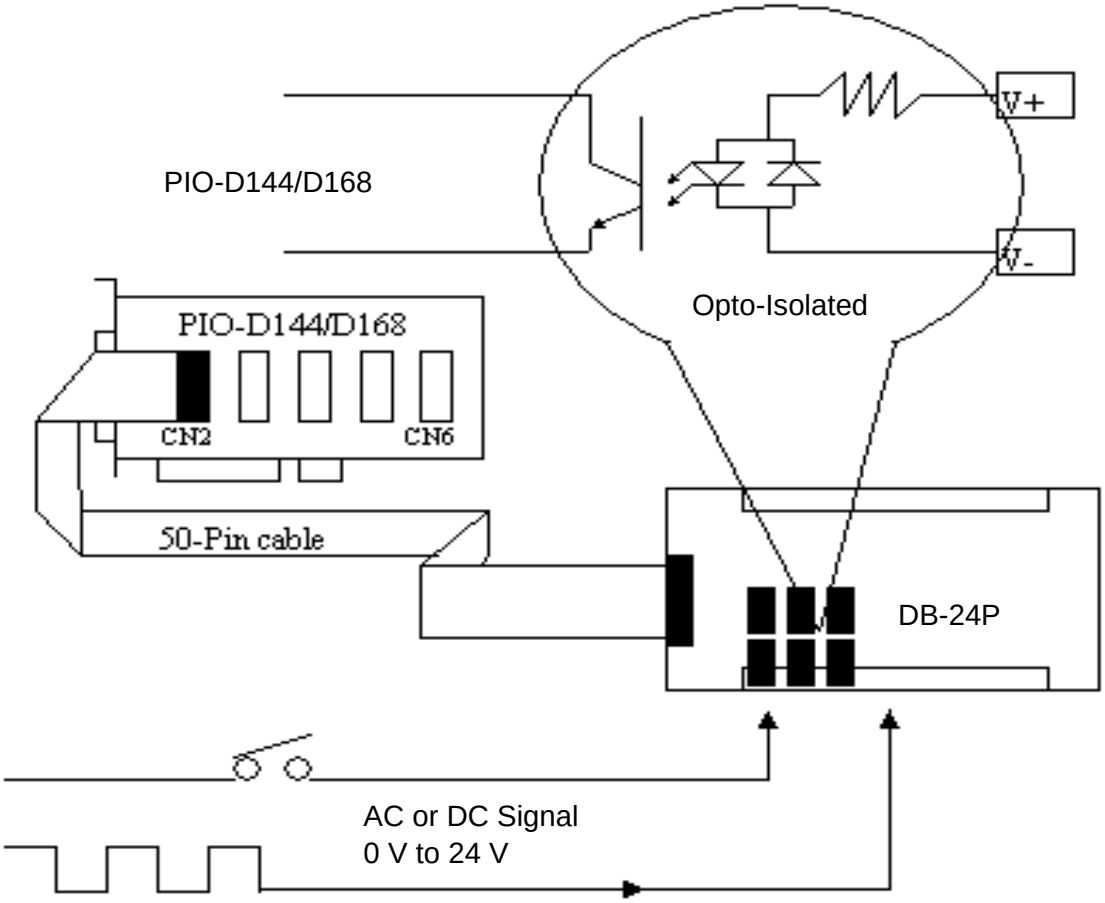

Figure 2.7

Table 2.4

	DB-24P	DB-24PD
50-pin flat-cable header	Yes	Yes
D-sub 37-pin header	No	Yes
Other specifications	Same	

2.7.6 DB-24R, DB-24RD Relay Board

The DB-24R, 24-channel relay output board, consists of 24 form-C relays for efficiently controlling the switch of the load by program. The relays are energized by applying 12 V/24 V voltage signal to the appropriated relay channel on the 50-pin flat-cable connector. There are 24 enunciator LEDs for each relay channel and the LED is light when their associated relay is activated. The control scheme is presented as below. Table 2.5 and 2.6 are the description of the daughter board for this application.

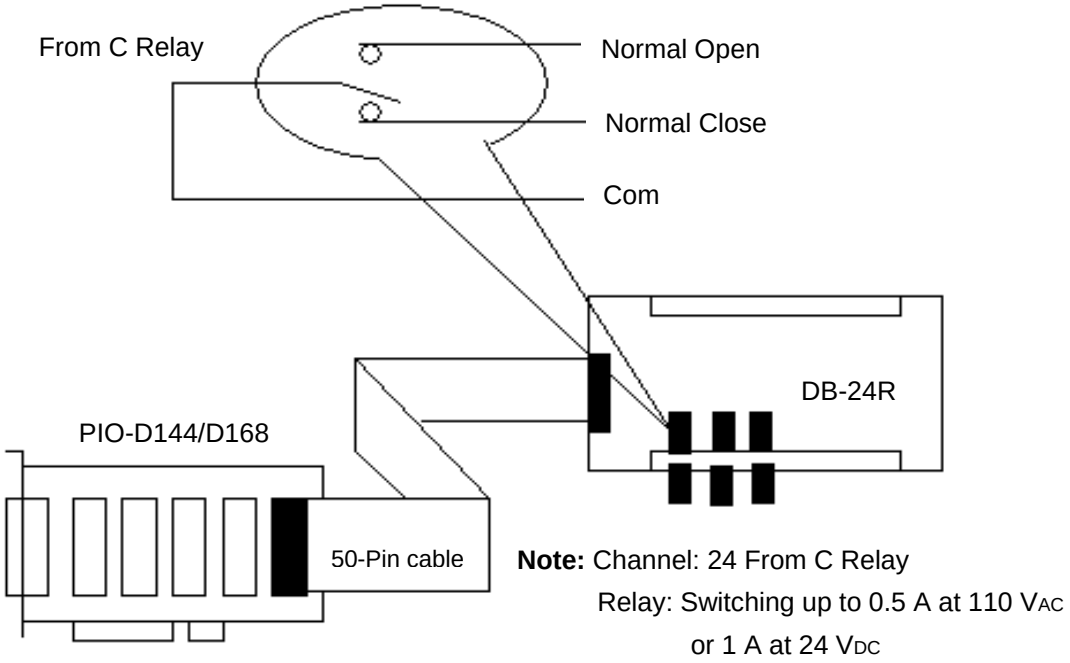

Figure 2.8

Table 2.5

	DB-24R	DB-24RD
50-pin flat-cable header	Yes	Yes
D-sub 37-pin header	No	Yes
Other specifications	Same	

Table 2.6

DB-24R, DB-24RD	24 × Relay (120 V, 0.5 A)
DB-24PR,DB-24PRD	24 × Power Relay (250 V, 5 A)
DB-24POR	24 × Photo MOS Relay (350 V, 01 A)
DB-24SSR	24 × SSR (250 V_{AC}, 4 A)
DB-24C	24 × O.C. (30 V, 100 mA)
DB-16P8R	16 × Relay (120 V, 0.5 A) + 8 × isolated input

2.7.7 DB-24PR, DB-24POR, DB-24C

Table 2.7

DB-24PR	24 × power relay, 5 A/250 V
DB-24POR	24 × Photo MOS relay, 0.1 A/350 V_{AC}
DB-24C	24 × open collector, 100 mA per channel, 30 V max.

The DB-24PR, 24-channel power relay output board, consists of 8 form-C and 16 form-A electromechanical relays for efficiently controlling the switch of the load by program. The contact of each relay can allow 5 A current load at 250 V_{AC}/30 V_{DC}. The relay is energized by applying a 5 voltage signal to the associate relay channel on the 20-pin flat-cable connector (just used 16 relays) or 50-pin flat-cable connector (OPTO-22 compatible, for DIO-24 series). 24 enunciator LEDs for indicating the status of for each relay and the corresponding LED is light when their associated relay is activated. To avoid overloading your PC's power supply, this board needs a +12 V_{DC} or +24 V_{DC} external power supply, as shown in figure 2.9.

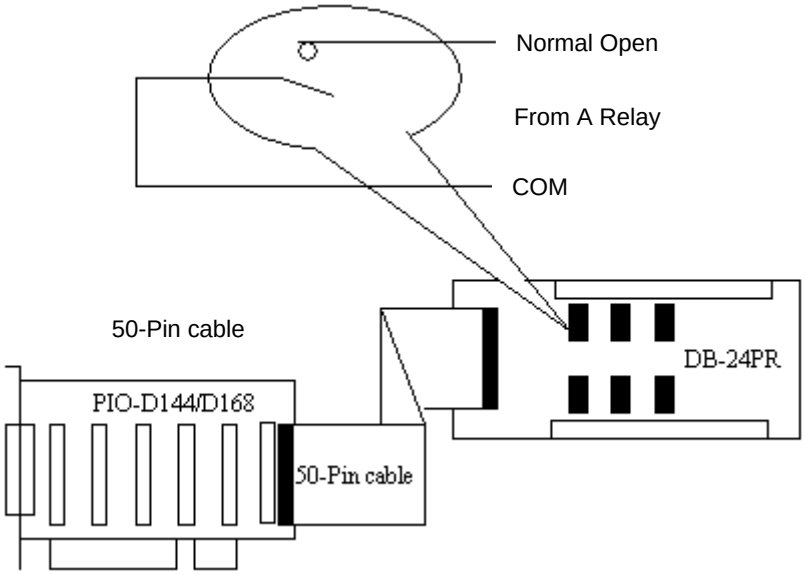

Figure 2.9

Note:

1. 50-Pin connector(OPTO-22 compatible), for DIO-24, DIO-48, DIO-144, PIO-D144, PIO-D96, PIO-D56, PIO-D48, PIO-D24,PIO-D168
2. 20-Pin connector for 16 channel digital output, A-82X, A-62X, DIO-64, ISO-DA16/DA8,
3. Channel: 16 Form A Relay, 8 Form C Relay.
4. Relay: switching up to 5 A at 110 V_{AC} / 5 A at 30 V_{DC}.

2.7.8 Daughter Board Comparison Table

Table 2.9 is the comparison table for the daughter application of PIO/PISO series cards.

Table 2.9

	20-pin flat-cable	50-pin flat-cable	D-sub 37-pin
DB-37	No	No	Yes
DN-37	No	No	Yes
ADP-37/PCI	No	Yes	Yes
ADP-50/PCI	No	Yes	No
DB-24P	No	Yes	No
DB-24PD	No	Yes	Yes
DB-16P8R	No	Yes	Yes
DB-24R	No	Yes	No
DB-24RD	No	Yes	Yes
DB-24C	Yes	Yes	Yes
Db-24PRD	No	Yes	Yes
DB-24POR	Yes	Yes	Yes
DB-24SSR	No	Yes	Yes

3. I/O Control Register

*3.1 H*ow to Find the I/O Address

The plug & play BIOS will assign a proper I/O address to every PIO/PISO series card in the power-on stage. The fixed IDs of PIO-D168/144 cards are given as following:

For PIO-D168(A) :

< PIO-D168A > :

- **Vendor ID = 0xE159**
- **Device ID = 0x0002**
- **Sub-vendor ID = 0x80**
- **Sub-device ID = 0x01**
- **Sub-aux ID = 0x50**

< PIO-D168 > :

- **Vendor ID = 0xE159**
- **Device ID = 0x0001**
- **Sub-vendor ID = 0x9880**
- **Sub-device ID = 0x01**
- **Sub-aux ID = 0x50**

For PIO-D144 :

< REV 1.0 ~ REV 3.0 > :

- **Vendor ID = 0xE159**
- **Device ID = 0x0002**
- **Sub-vendor ID = 0x80**
- **Sub-device ID = 0x01**
- **Sub-aux ID = 0x00**

< REV 4.0 or above > :

- **Vendor ID = 0xE159**
- **Device ID = 0x0001**
- **Sub-vendor ID = 0x5C80**
- **Sub-device ID = 0x01**
- **Sub-aux ID = 0x00**

The utility program, PIO_PISO.EXE, will detect and present all information of PIO/PISO cards installed in this PC, as shown in following figure. Besides, for how to identify the PIO series cards of ICPDAS data acquisition board by the sub-vender, sub-device and sub-Aux ID is given in table 3-1.

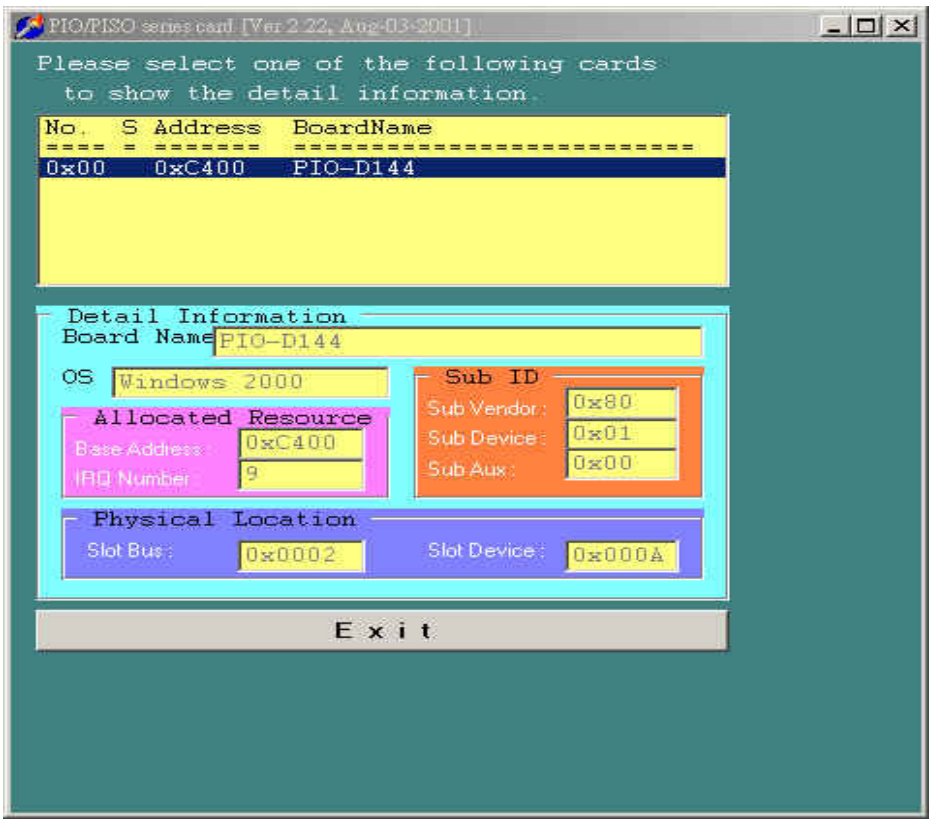
PIO/PISO series card [Ver 2.22, Aug-03-2001]
Please select one of the following cards
to show the detail information.
No. S Address BoardName
0x00 0xC400 PIO-D144
Detail Information
Board Name PIO-D144
OS Windows 2000
Allocated Resource
Base Address: 0xC400
IRQ Number: 9
Sub ID
Sub Vendor: 0x80
Sub Device: 0x01
Sub Aux: 0x00
Physical Location
Slot Bus: 0x0002
Slot Device: 0x000A
Exit

Figure 3.1

Table 3-1

PIO/PISO series card	Description	Sub_Sendor	Sub_Device	Sub_AUX
PIO-D168	168 * DIO	9880	01	50
PIO-D168A	168 * DIO	80	01	50
PIO-D144(REV4.0)	144 * D/I/O	80 (5C80)	01	00
PIO-D96	96 * D/I/O	80	01	10
PIO-D64 (REV2.0)	64 * D/I/O	80 (4080)	01	20
PIO-D56	24 * D/I/O + 16 * D/I+16*D/O	80	01	40
PIO-D48	48 * D/I/O	80	01	30
PIO-D24	24 * D/I/O	80	01	40
PIO-821	Multi-function	80	03	10
PIO-DA16	16 * D/A	80	04	00
PIO-DA8	8 * D/A	80	04	00
PIO-DA4	4 * D/A	80	04	00
PISO-C64	64 * isolated D/O (**Current sinking**)	80	08	00
PISO-A64	64 * isolated D/O **(Current sourcing)**	80	08	50
PISO-P64	64 * isolated D/I	80	08	10
PISO-P32C32	32* isolated D/O (**Current sinking**) + 32* isolated D/I	80	08	20
PISO-P32A32	32*isolated DO (**Current sourcing**) + 32* isolated D/I	80	08	70
PISO-P8R8	8* isolated D/I + 8 * 220 V relay	80	08	30
PISO-P8SSR8AC	8* isolated D/I + 8 * SSR /AC	80	08	30
PISO-P8SSR8DC	8* isolated D/I + 8 * SSR /DC	80	08	30
PISO-730	16*DI + 16*D/O + 16* isolated D/I + 16*isolated D/O (**Current sinking**)	80	08	40
PISO-730A	16*DI + 16*D/O + 16* isolated D/I + 16*isolated D/O (**Current sourcing**)	80	08	80
PISO-813	32 * isolated A/D	80	0A	00
PISO-DA2	2 * isolated D/A	80	0B	00

Note: If the board has different version, it may has different Sub IDs. But no matter which version of the board you select, we offer the same function calls.

3.2 The Assignment of I/O Address

The Plug & Play BIOS will assign the proper I/O address to PIO/PISO series card. If there is only one PIO/PISO board, the user can identify the board as card_0. If there are two PIO/PISO boards in the system, the user will be very difficult to identify which board is card_0. The software driver can support the maximum 16 boards. Therefore, the user can install 16 boards of PIO/PSIO series cards in one PC system. For how to find and identify the card_0 & card_1 is demonstrated as below:

The simplest way to identify which card is card_0 is to use wSlotBus & wSlotDevice as follows:

1. Remove all PIO-D144/D168 from this PC
2. Install one PIO-D144/D168 into the PC's PCI_slot1, run PIO_PISO.EXE. Then record the wSlotBus1 and wSlotDevice1
3. Remove all PIO-D144/168 from this PC
4. Install one PIO-D144/168 into the PC's PCI_slot2 and run PIO_PISO.EXE. Then record the wSlotBus2 and wSlotDevice2 information.
5. Repeat (3) & (4) for all PCI_slot and record all information of wSlotBus and wSlotDevice.
6. The records may be as follows:

Table 3-2

PC's PCI slot	WslotBus	WslotDevice
Slot_1	**0**	**0x07**
Slot_2	**0**	**0x08**
Slot_3	**0**	**0x09**
Slot_4	**0**	**0x0A**
PCI-BRIDGE		
Slot_5	**1**	**0x0A**
Slot_6	**1**	**0x08**
Slot_7	**1**	**0x09**
Slot_8	**1**	**0x07**

The above procedure records all information of wSlotBus and wSlotDevice in this PC. These values will be mapped to this PC's physical slot. And this

mapping will not be changed for any PIO/PISO cards. Therefore, this information can be used to identify the specified PIO/PISO card by following steps:

Step1: Using the information of wSlotBus and wSlotDevice in table 3-2

Step2: Input board number into funtion PIO_GetConfigAddressSpace(…) to get the specified card's information, especially wSlotBus and wSlotDevice

Step3: The user can identify the specified PIO/PISO card by comparing the data of the wSlotBus & wSlotDevice in step1 and step2.

Note that normally the card installed in slot 0 is the card0 and card installed in slot1 is the card1 for PIO/PISO series cards.

3.3 The I/O Address Map

The I/O address of PIO/PISO series card is automatically assigned by the main board ROM BIOS. The I/O address can also be re-assigned by user. **It is strongly recommended not to change the I/O address by user. The Plug & Play BIOS will assign proper I/O address to each PIO/PISO series card very well**. The I/O addresses of PIO-D144/D168 are given as follows, which are based on the base address of each card.

Table 3-3

Address	Read	Write
WBase+0	RESET\ control register	RESET\ control register
WBase+2	Aux control register	Aux control register
WBase+3	Aux data register	Aux data register
WBase+5	INT mask control register	INT mask control register
WBase+7	Aux pin status register	Aux pin status register
WBase+0x2a	INT polarity control register	INT polarity control register
WBase+0xc0	Read 8-bit data from D/I port	Write 8-bit data to D/O port
WBase+0xc4	Reserved	Select the active I/O port
WBase+0xc8	Reserved	I/O Port 0-5 direction control
WBase+0xcc	Reserved	I/O Port 6-11 direction control
WBase+0xd0	Reserved	I/O Port 12-17 direction control
WBase+0xd4	Reserved	I/O Port 18-20 direction control

Note: Wbase+0xd4 is only for PIO-D168.

3.3.1 RESET\ Control Register

(Read/Write): wBase+0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	RESET\

When the PC is first power-on, the RESET\ signal is in Low-state. **This will disable all D/I/O operations.** The user has to set the RESET\ signal to High-state before any D/I/O command application.

```
outp(wBase,1);      /*  RESET\=High → all D/I/O are enable now */
outp(wBase,0);      /*  RESET\=Low → all D/I/O are disable now */
```

3.3.2 AUX Control Register

(Read/Write): wBase+2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Aux?=0→ this Aux is used as a D/I
Aux?=1→ this Aux is used as a D/O

When the PC is first power-on, all Aux signal are in Low-state. All Aux are designed as D/I for all PIO/PISO series.

3.3.3 AUX data Register

(Read/Write): wBase+3

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

When the Aux is used as D/O, the output state is controlled by this register. This register is designed for feature extension. Therefore, do not use this register.

3.3.4 INT Mask Control Register

(Read/Write): wBase+5

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	EN3	EN2	EN1	EN0

EN0=0→ Disable P2C0 of CN1 as a interrupt signal (Default).
EN0=1→ Enable P2C0 of CN1 as a interrupt signal

```
outp(wBase+5,0);        /* Disable interrupt                          */
outp(wBase+5,1);        /* Enable interrupt P2C0                      */
outp(wBase+5,0x0f);     /* Enable interrupt P2C0,P2C1,P2C2,P2C3 */
```

3.3.5 Aux Status Register

(Read/Write): wBase+7

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Aux7	Aux6	Aux5	Aux4	Aux3	Aux2	Aux1	Aux0

Aux0=P2C0, Aux1=P2C1, Aux2=P2C2, Aux3=P2C3, Aux7~4=Aux-ID. Refer to DEMO5.C for more information. The Aux0~3 are used as interrupt source. The interrupt service routine has to read this register for interrupt source identification. Refer to Sec. 2.6 for more information.

3.3.6 Interrupt Polarity Control Register

(Read/Write): wBase+0x2A

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	INV3	INV2	INV1	INV0

This register provides a function to control invert or non-invert of the interrupt signal source. The detail application example is shown in below.

- INV0=1→ select the non-inverted signal from P2C0
- INV0=0→ select the inverted signal from P2C0

```
outp(wBase+0x2a,0);       /* select the inverted input P2C0/1/2/3     */
outp(wBase+0x2a,0x0f);    /* select the non-inverted input of P2C0/1/2/3      */
outp(wBase+0x2a,1);       /* select the non-inverted input of P2C0            */
                          /* select the inverted input P2C1/2/3       */
outp(wBase+0x2a,3);       /* select the non-inverted input of P2C0/1          */
                          /* select the inverted input P2C2/3         */
```

Refer to Sec. 2.6 and DEMO5.C for more information.

3.3.7 Read/Write 8-bit data Register

(Read/Write): wBase+0xc0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

There are 18/21 8-bit I/O ports in the PIO-D144/D168. Every I/O port can be configured as D/I or D/O port. User can send/receive digital data to/from this register for digital input or output. Note that all ports are used as D/I port when the PC is first power-on.

```
outp(wBase+0xc0,Val);          /* write to D/O port   */
Val=inp(wBase+0xc0);           /* read from D/I port */
```

Note: Make sure the I/O port configuration (DI or DO) before read/write the data register.(refer to sec.3.3.9)

3.3.8 Active I/O Port Control Register

(Read/Write): wBase+0xc4

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

There are 18/21 8-bit I/O ports in the PIO-D144/D168. Only one I/O port can be activated at the same time. The following example is demonstration for how to activate the port of the PIO series card.

```
outp(wBase+0xc4,0);        /*  I/O port0 is active now   */

outp(wBase+0xc4,1);        /*  I/O port1 is active now   */

outp(wBase+0xc4,17);       /*  I/O port17 is active now */

outp(wBase+0xc4,20);       /*  I/O port20 is active now */
```

3.3.9 I/O Selection Control Register

(Write): wBase+0xc8

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	Port5	Port4	Port3	Port2	Port1	Port0

(Write): wBase+0xcc

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	Port11	Port10	Port9	Port8	Port7	Port6

(Write): wBase+0xd0

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	Port17	Port16	Port15	Port14	Port13	Port12

(Write): wBase+0xd4

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	Port20	Port19	Port18

Port?=1→ This port is used as a D/I port
Port?=0→ This port is used as a D/O port

These registers provide the function for configuration digital input/output port of the PIO/PISO series cards. Every I/O port can be programmed as D/I or D/O port. Note that all ports are used as D/I port when the PC is first power-on. The I/O port location is given as follows and configuration example is also presented in below.

Table 3-4

Connector of PIO-D144/D168	PA0 ~ PA7	PB0 ~ PB7	PC0 ~ PC7
CN1	Port0	Port1	Port2
CN2	Port3	Port4	Port5
CN3	Port6	Port7	Port8
CN4	Port9	Port10	Port11
CN5	Port12	Port13	Port14
CN6	Port15	Port16	Port17
CN7 (PIO-D168 Only)	Port18	Port19	Port20

```
outportb(wBase+0xc8,0);          /*  Port0 to Port5 are all D/O port      */

outportb(wBase+0xcc,0x3f);       /*  Port6 to Port11 are all D/I port     */

outportb(wBase+0xd0,0x38);       /*  Port12 to Port14 are all D/O port  */
                                 /*  Port15 to Port17 are all D/I port    */
```

4. Software Installation

The PIO-D144/D168 can be used in DOS and Windows 98/Me/NT/2000/XP. For Windows O.S, the recommended installation steps are given in Sec 4.1 ~ 4.2

4.1 Software Installing Procedure

Step 1: Insert the companion CD into the CD-ROM driver and wait a few seconds until the installation program starts automatically. If it cannot be started automatically for some reasons, please double-click the file \NAPDOS\AUTO32.EXE in this CD.

Step 2: Click the item: Install Toolkits (Software) / Manuals.

Step 3: Click the item: PCI Bus DAQ Card.

Step 4: Click PIO-DIO.

Step 5: Click “install Toolkit for Windows 98 (Or Me, NT, 2000, XP)”.

Then, the InstallShield will start the driver installation process to copy the related material to the indicated directory and register the driver on your computer. The driver target directory is as below for different system.

Windows NT/2000/XP :
The PIODIO.DLL will be copied into C:\WINNT\SYSTEM32
The NAPWNT.SYS and PIO.SYS will be copied into
C:\WINNT\SYSTEM32\DRIVERS

Windows 98/Me :
The PIODIO.DLL,and PIODIO.Vxd will be copied into C:\Windows\SYSTEM

4.2 PnP Driver Installation

After installing the hardware (PIO-D144/D168) and power on your PC, Windows 98/Me/2000/XP can find a PCI card device and ask user to provide a PIODIO.inf to install hardware driver on the computer. If user has trouble to precede this process, please refer to PnPinstall.pdf for more information.

5. DLL Function Description

The DLL driver is the collection of function calls of the PIO-DIO cards for Windows 98/Me/NT/2000/XP system. The application structure is presented as following figure. The user application program developed by designate tools like VB, Delphi and Borland C^{++} Builder can call PIODIO.DLL driver in user mode. And then DLL driver will call the PIO.sys to access the hardware system.

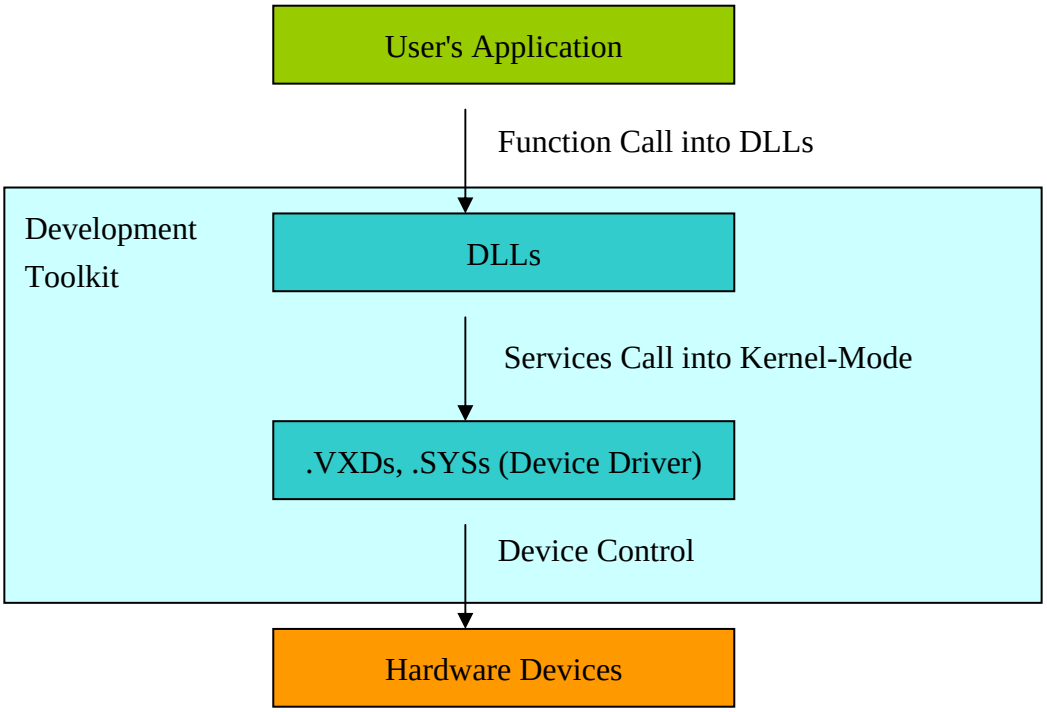

Figure 5.1

5.1 Table of ErrorCode and ErrorString

Table 5.1

Error Code	Error ID	Error String
0	PIODIO_NoError	OK (No error !)
1	PIODIO_DriverOpenError	Device driver can't be opened
2	PIODIO_DriverNoOpen	Users have to call the DriverInit function firstly
3	PIODIO_GetDriverVersionError	Get driver version error
4	PIODIO_InstallIrqError	Install IRQ Error
5	PIODIO_ClearIntCountError	Clear counter value Error
6	PIODIO_GetIntCountError	Get counter of interrput error
7	PIODIO_RemoveIrqError	Remove IRQ Error
8	PIODIO_FindBoardError	Can not find board
9	PIODIO_ExceedBoardNumber	The Max. boards is: 8
10	PIODIO_ResetError	Can't reset interrupt count
11	PIODIO_IrqMaskError	Irq-Mask is 1,2,4,8 or 1 to 0xF
12	PIODIO_ActiveModeError	Active-Mode is 1,2 or 1 to 3
13	PIODIO_GetActiveFlagError	Can’t get interrupt active flag
14	PIODIO_ActiveFlagEndOfQueue	The flag queue is empty

5.2 Function Descriptions

All of the functions provided for PIO-DI44/D168 are listed as below and the detail information for every function will be presented in the following section. However, in order to make the description simplify and clearly, the attribute of the input and output parameter of the function is indicated as [input] and [output] respectively, as shown in following table.

Table 5.2

Keyword	Setting parameter by user before calling this function ?	Get the data/value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

Table 5.3

Function Definition	
float	PIODIO_FloatSub(float fA, float fB);
short	PIODIO_ShortSub(short nA, short nB);
WORD	PIODIO_GetDllVersion(void);
WORD	PIODIO_DriverInit(void);
void	PIODIO_DriverClose(void);
WORD	PIODIO_SearchCard(WORD *wBoards, DWORD dwPIOCardID);
WORD	PIODIO_GetDriverVersion(WORD *wDriverVersion);
WORD	PIODIO_GetConfigAddressSpace(WORD wBoardNo, DWORD *wAddrBase, WORD *wIrqNo, WORD *wSubVendor, WORD *wSubDevice,WORD *wSubAux, WORD *wSlotBus,WORD *wSlotDevice);
WORD	PIODIO_ActiveBoard(WORD wBoardNo);
WORD	PIODIO_WhichBoardActive(void);
void	PIODIO_OutputWord(DWORD wPortAddress, DWORD wOutData);
void	PIODIO_OutputByte(DWORD wPortAddr, WORD bOutputValue);
DWORD	PIODIO_InputWord(DWORD wPortAddress);
WORD	PIODIO_InputByte(DWORD wPortAddr);
WORD	PIODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent, WORD wInterruptSource, WORD wActiveMode);
WORD	PIODIO_IntRemove(void);
WORD	PIODIO_IntResetCount(void);
WORD	PIODIO_IntGetCount(DWORD *dwIntCount);

5.3 FUNCTIONS OF TEST

5.3.1 PIODIO_GetDllVersion

- **Description:**

  To get the version number of PIODIO.DLL driver

- **Syntax:**

  WORD PIODIO_GetDllVersion(Void)

- **Parameter:**

  None

- **Return:**

  200(hex) for version 2.00

5.3.2 PIODIO_ShortSub

- **Description:**
  To perform the subtraction as nA - nB in short data type. This function is provided for testing DLL linkage purpose.
- **Syntax:**
  short PIODIO_ShortSub(short nA, short nB)
- **Parameter:**
  nA :[Input] 2 bytes short data type value
  nB :[Input] 2 bytes short data type value
- **Return:**
  The value of nA – nB

5.3.3 PIODIO_FloatSub

- **Description:**
  To perform the subtraction as fA - fB in float data type. This function is provided for testing DLL linkage purpose.
- **Syntax:**
  float PIODIO_FloatSub(float fA, float fB)
- **Parameter:**
  fA : [Input] 4 bytes floating point value
  fB : [Input] 4 bytes floating point value
- **Return:**
  The value of fA - fB

5.4 Digital I/O FUNCTIONS

5.4.1 PIODIO_OutputByte

- **Description :**

  This subroutine sends the 8 bits data to the specified I/O port.

- **Syntax :**

  void PIODIO_OutputByte(DWORD wPortAddr, WORD bOutputVal);

- **Parameter :**

  WPortAddr : [Input] I/O port addresses, please refer to function PIODIO_GetConfigAddressSpace. Only the low WORD is valid.

  bOutputVal: [Input] 8 bit data send to I/O port. Only the low BYTE is valid.

- **Return:**

  None

5.4.2 PIODIO_InputByte

- **Description :**

  This subroutine reads the 8 bits data from the specified I/O port.

- **Syntax :**

  WORD PIODIO_InputByte(DWORD wPortAddr);

- **Parameter :**

  wPortAddr : [Input] I/O port addresses, please refer to function PIODIO_GetConfigAddressSpace(). Only the low WORD is valid.

- **Return:**

  16 bits data with the leading 8 bits are all 0.(Only the low BYTE is valid.)

5.4.3 PIODIO_OutputWord

- **Description :**

  This subroutine sends the 16 bits data to the specified I/O port.

- **Syntax :**

  void PIODIO_OutputWord(DWORD wPortAddr, DWORD wOutputVal);

- **Parameter :**

  WPortAddr : [Input] I/O port addresses, please refer to function PIODIO_GetConfigAddressSpace().
  Only the low WORD is valid.

  WOutputVal : [Input] 16 bit data send to I/O port.
  Only the low WORD is valid.

- **Return:**

  None

5.4.4 PIODIO_InputWord

- **Description :**

  This subroutine reads the 16 bits data from the specified I/O port.

- **Syntax :**

  DWORD PIODIO_InputWord(DWORD wPortAddr);

- **Parameter :**

  wPortAddr : [Input] I/O port addresses, please refer to function PIODIO_GetConfigAddressSpace().
  Only the low WORD is valid.

- **Return:**

  16 bit data. Only the low WORD is valid.

5.5 Driver Relative Functions

5.5.1 PIODIO_GetDriverVersion

- **Description :**
  This subroutine reads the version number information from PIODIO driver.
- **Syntax :**
  WORD PIODIO_GetDriverVersion(WORD *wDriverVersion);
- **Parameter :**
  wDriverVersion : [Output] address of wDriverVersion
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.5.2 PIODIO_DriverInit

- **Description :**
  This subroutine opens the PIODIO driver and allocates the computer resource for the device. This function must be called once before applying other PIODIO functions.
- **Syntax :**
  WORD PIODIO_DriverInit();
- **Parameter :**
  None
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.5.3 PIODIO_SearchCard

- **Description :**

  This subroutine can search the card and get total boards. This function must be called once before applying other PIODIO functions.

- **Syntax :**

  WORD PIODIO_SearchCard(WORD *wBoards, DWORD dwPIOCardID);

- **Parameter :**

  wBoards : [Output] Get the total boards.
  DwPIOCardID : [Input] Sub id of PIODIO card. Please refer table3.1

  NOTE :

  Different version PIO-D168/144 boards may have different Sub IDs. This function will find the total board of PIO-D168/144 including all versions, no matter what version Sub ID you input. Following is the example demonstration:

  wRtn=PIODIO_SearchCard(&wBoards, 0x800100);

  you will get the total numbers of PIO-D144 boards including all versions in PC.

- **Return:**

  Please refer to "Section 5.1 Error Code".

5.5.4 PIODIO_DriverClose

- **Description :**

  This subroutine closes the PIODIO Driver and releases the resource from computer device resource. This function must be called once before exiting the user's application.

- **Syntax :**

  void PIODIO_DriverClose();

- **Parameter :**

  None

- **Return:**

  None

5.5.5 PIODIO_GetConfigAddressSpace

- **Description :**

  Obtain the I/O address and other information of PIODIO board.

- **Syntax :**

  WORD PIODIO_GetConfigAddressSpace(WORD wBoardNo,
  DWORD *wAddrBase, WORD *wIrqNo, WORD *wSubVendor,
  WORD *wSubDevice, WORD *wSubAux, WORD *wSlotBus,
  WORD *wSlotDevice);

- **Parameter :**

  wBoardNo : [Input] PIODIO board number
  wAddrBase : [Output] The base address of PIODIO board.
  Only the low WORD is valid.
  wIrqNo : [Output] The IRQ number that the PIODIO board using.
  wSubVendor : [Output] Sub Vendor ID.
  wSubDevice : [Output] Sub Device ID.
  wSubAux : [Output] Sub Aux ID.
  wSlotBus : [Output] Slot Bus number.
  wSlotDevice : [Output] Slot Device ID.

- **Return:**

  Please refer to "Section 5.1 Error Code".

5.6 INTERRUPT FUNCTION

5.6.1 PIODIO_IntResetCount

- **Description:**
  This function will clear the counter value on the device driver for the interrupt.
- **Syntax:**
  WORD PIODIO_IntResetCount(void);
- **Parameter:**
  None
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.6.2 PIODIO_IntGetCount

- **Description:**
  This subroutine will read the **dwIntCount** defined in device driver.
- **Syntax :**
  WORD PIODIO_IntGetCount(DWORD *dwIntCount);
- **Parameter:**
  dwIntCount : [Output] Address of dwIntCount, which will stores the counter value of interrupt.
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.6.3 PIODIO_IntInstall

- **Description:**

  This subroutine installs the IRQ service routine.

- **Syntax:**

  WORD PIODIO_IntInstall(WORD wBoardNo, HANDLE *hEvent,
  WORD wInterruptSource, WORD wActiveMode);

- **Parameter:**

  wBoardNo : [Input] Which board to be used.

  hEvent : [Input] Address of a Event handle. The user's program must call the Windows API function "CreateEvent()" to create the event-object.

  wInterruptSource : [Input] What the Interrupt-Source to be used ? Please refer to the following table.

Table 5.4

Card No.	wInterruptSource	Description
PIO-D48	0	PC3/PC7 from Port-2
	1	PC3/PC7 from Port-5
	2	Cout0
	3	Cout2
PIO-D56 PIO-D24	0	PC0
	1	PC1
	2	PC2
	3	PC3
PIO-D64	0	EXTIRQ
	1	EVTIRQ
	2	TMRIRQ
PIO-D96	0	P2C0
	1	P5C0
	2	P8C0
	3	P11C0
PIO-D144/D168	0	P2C0
	1	P2C1
	2	P2C2
	3	P2C3

  wActiveMode : [Input] When to trigger the interrupt ?
  0 → PIODIO_ActiveLow
  1 → PIODIO_ActiveHigh

- **Return:**

  Please refer to "Section 5.1 Error Code".

5.6.4 PIODIO_IntRemove

- **Description:**
  This subroutine removes the IRQ service routine.
- **Syntax:**
  WORD PIODIO_IntRemove(void);
- **Parameter:**
  None
- **Return:**
  Please refer to "Section 5.1 Error Code".

5.6.5 Architecture of Interrupt mode

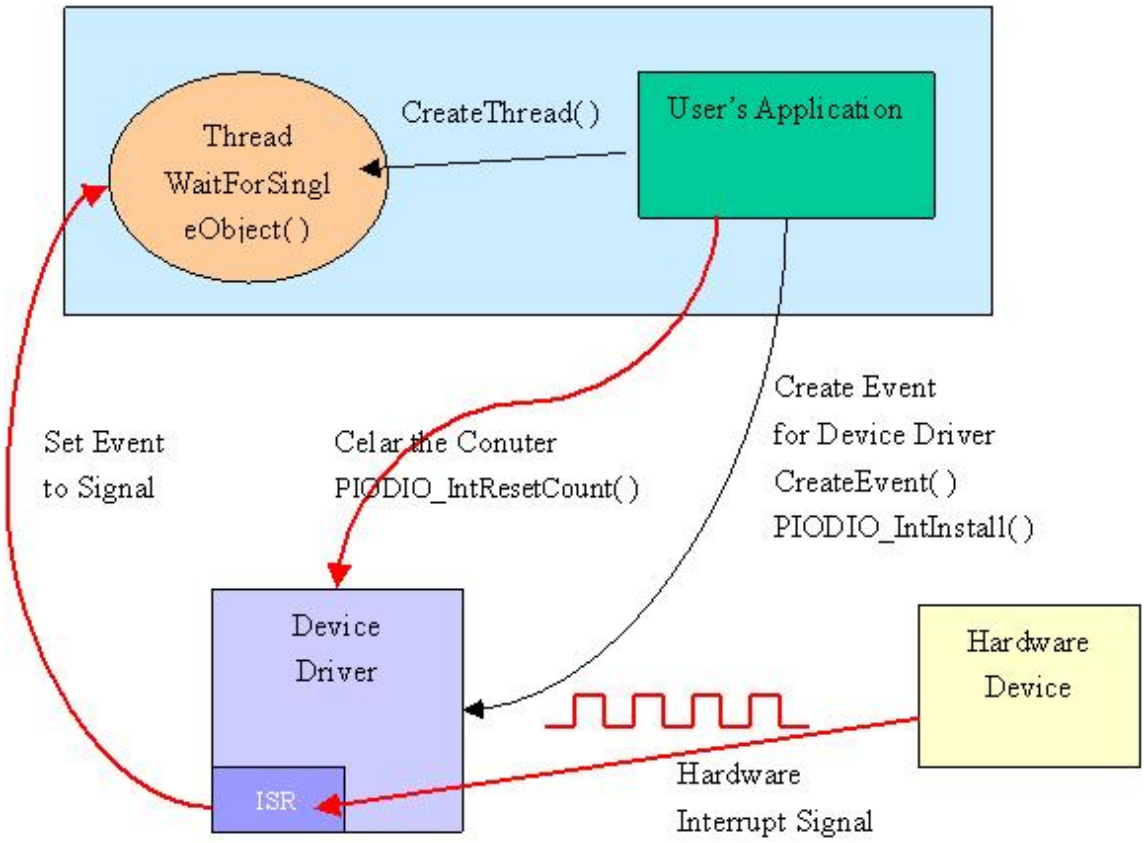

Figure 5.2

5.6.6 Program Architecture

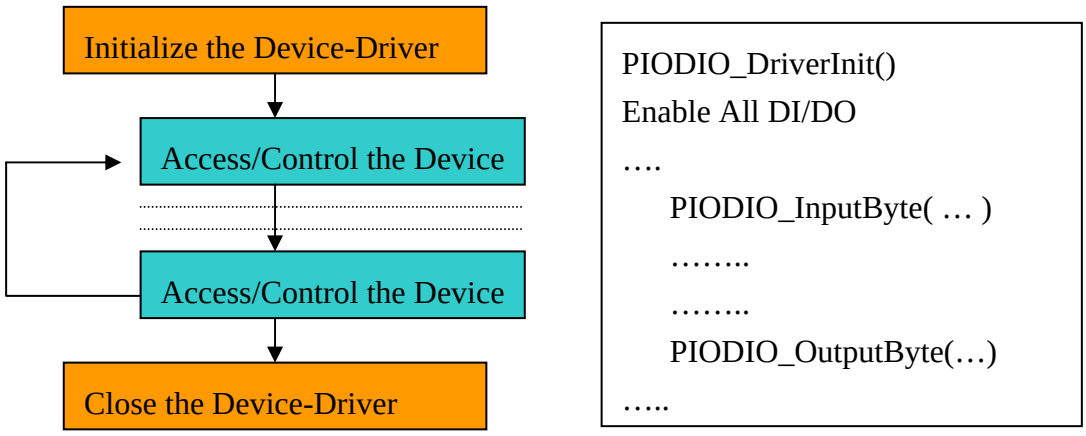

Figure 5.3

6. Demo Programs for Windows

All of demo programs will not work normally if DLL driver would not be installed correctly. During the installation process of DLL driver, the install-shields will register the correct kernel driver to the operation system and copy the DLL driver and demo programs to the correct position based on the driver software package you have selected (Win98,Me,NT,win2000,XP). After driver installation, the related demo programs and development library and declaration header files for different development environments are presented as follows.

```
|--\Demo                          → demo program
   |--\BCB3                       → for Borland C++ Builder 3
   |    |--\PIODIO.H              → Header file
   |       \ PIODIO.LIB           → Linkage library for BCB only
   |
   |--\Delphi3                    → for Delphi3
   |    |--\ PIODIO.PAS           → Declaration file
   |
   |--\VB6                        → for Visual Basic 6
      |--\ PIODIO.BAS             →Declaration file
```

The list of demo programs :

DO : D/O of CN1
DIO : DI/O of CN5 and CN6
DIO2 : DI/O of all Ports
INT : Interrupt of P2C0

6.1 *Digital Output of CN1(Port0~Port2)*

This demo program is used to check the digital output status of CN1. You can connect CN1 to a DB-24C, and you can check the LED of DB-24C to find the digital output status of CN1.

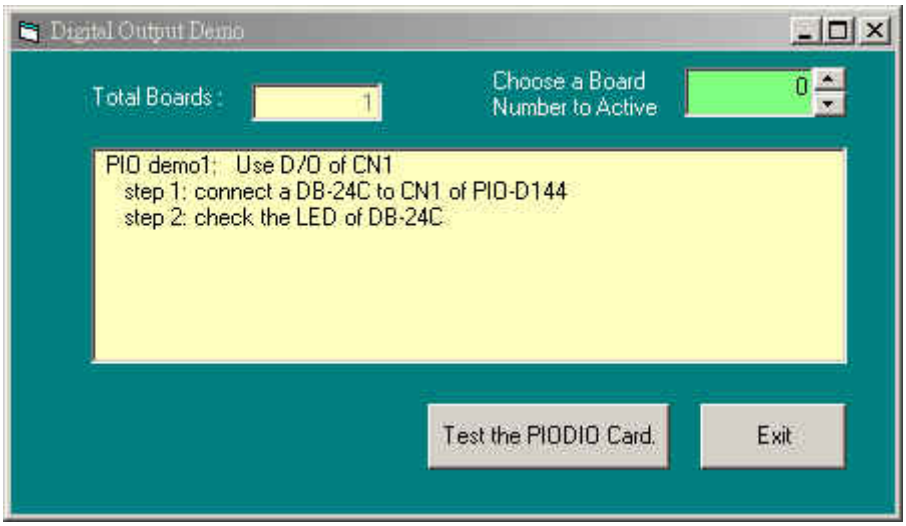

Figure 6.1

6.2 *Digital input /output of CN5 and CN6*

This demo program is used to check the digital output and digital input status of CN5 and CN6.

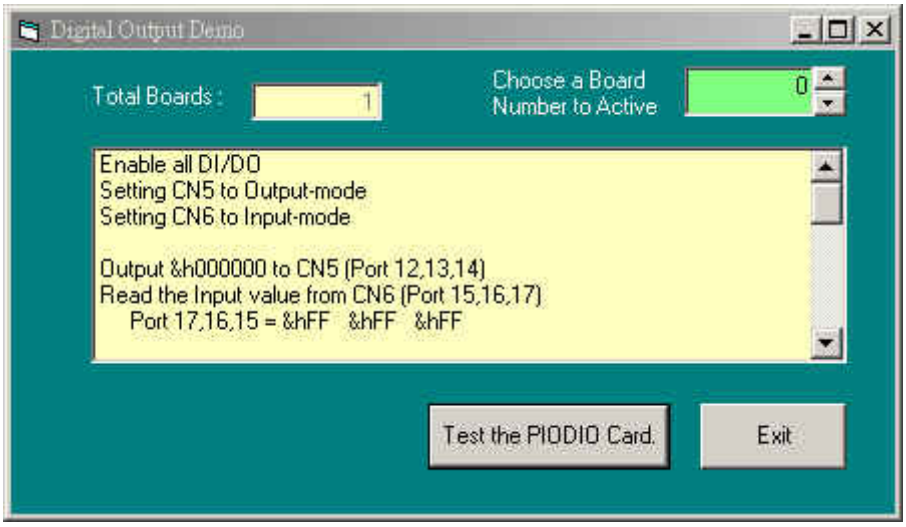

Figure 6.2

6.3 *Digital output / input of all Ports (Port0~20)*

In this demo program, you can read digital input status from port1 to port20 by selecting “Input Mode” and “Select Port”. And you can write a digital output value to the I/O port by selecting “Output Mode” and ”Select Port”. Then click the “Bits Status” to output the value.

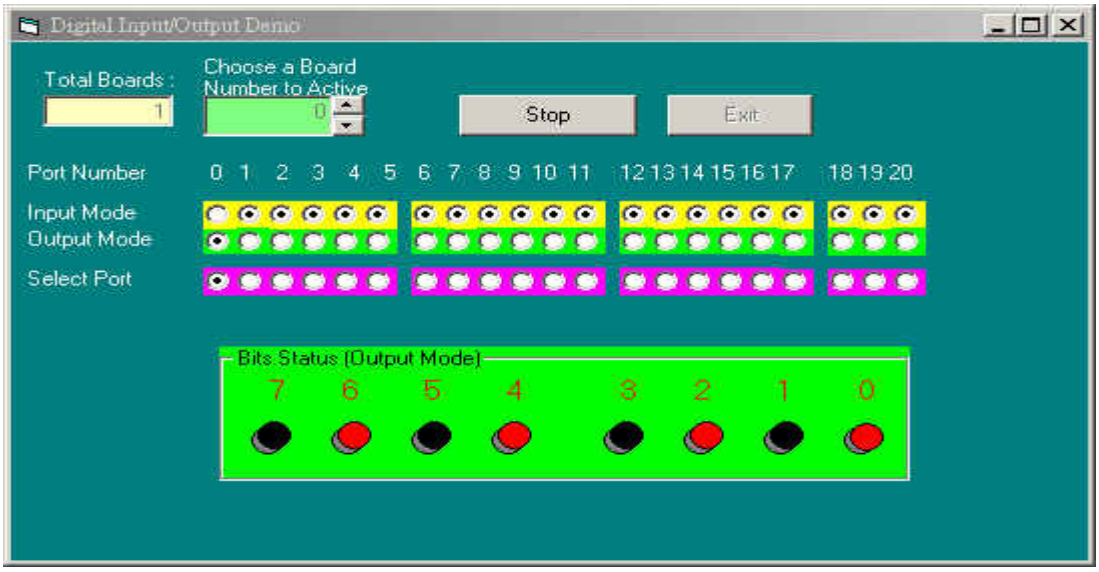

Figure 6.3

6.4 *Interrupt of P2C0*

This demo program uses port0 as digital output port and P2C0 as interrupt source. Then port0 will output a high and low signal repeatedly according to the trigger of the interrupt source.

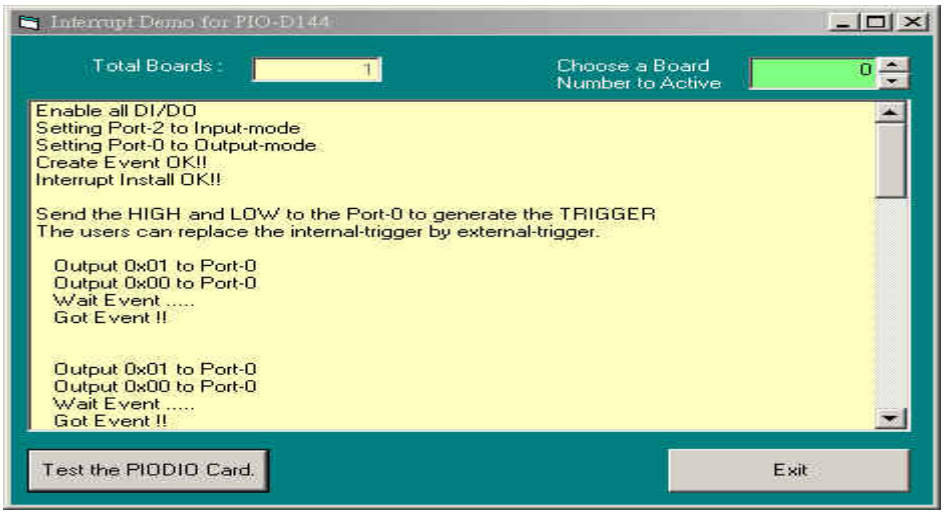

Figure 6.4

Appendix

Appendix A. Related DOS Software

A-1 Where is the related software

The related DOS software and demos in the CD is given as following:

- \TC*.* → for Turbo C 2.xx or above
- \MSC*.* → for MSC 5.xx or above
- \BC*.* → for BC 3.xx or above

- \TC\LIB*.* → for TC library
- \TC\DEMO*.* → for TC demo program
- \TC\DIAG*.* → for TC diagnostic program

- \TC\LIB\PIO.H → TC declaration file
- \TC\\LIB\TCPIO_L.LIB → TC large model library file
- \TC\\LIB\TCPIO_H.LIB → TC huge model library file

- \MSC\LIB\PIO.H → MSC declaration file
- \MSC\LIB\MSCPIO_L.LIB → MSC large model library file
- \MSC\\LIB\MSCPIO_H.LIB → MSC huge model library file

- \BC\LIB\PIO.H → BC declaration file
- \BC\LIB\BCPIO_L.LIB → BC large model library file
- \BC\\LIB\BCPIO_H.LIB → BC huge model library file

The list of demo programs :

DEMO1 : D/O of CN1
DEMO2 : D/O of CN1 ~ CN6
DEMO3 : Interrupt of P2C0 (Initial low & active high)
DEMO4 : Interrupt of P2C0 (Initial high & active low)
DEMO5 : 4 interrupt sources

A-2 DOS LIB Function

A-2-1 Table of ErrorCode and ErrorString

Table A.1 ErrorCode and ErrorString

Error Code	Error ID	Error String
0	NoError	OK ! No Error!
1	DriverHandleError	Device driver opened error
2	DriverCallError	Got the error while calling the driver functions
3	FindBoardError	Can't find the board on the system
4	TimeOut	Timeout
5	ExceedBoardNumber	Invalidate board number (Valid range: 0 to TotalBoards -1)
6	NotFoundBoard	Can't detect the board on the system

A-2-2 PIO_DriverInit

- **Description :**

  This function can detect all PIO/PISO series card in the system. It is implemented based on the PCI Plug & Play mechanism-1. It will find all PIO/PISO series cards installed in this system and save all their resource in the library.

- **Syntax :**

  WORD PIO_DriverInit(WORD *wBoards, WORD wSubVendorID,
  WORD wSubDeviceID,WORD wSubAuxID)

- **Parameter :**

  WBoards : [Output] Number of boards found in this PC
  wSubVendor : [Input] SubVendor ID of the board
  wSubDevice : [Input] SubDevice ID of the board
  wSubAux : [Input] SubAux ID of the board

- **Return:**

  Please refer to " Table A.1".

A-2-3 PIO_GetConfigAddressSpace

- **Description :**

  The user can use this function to save resource of all PIO/PISO cards installed in this system. Then the application program can control all functions of PIO/PISO series card directly.

- **Syntax :**

  WORD PIO_GetConfigAddressSpace(wBoardNo,*wBase,*wIrq,
  wSubVendor, *wSubDevice,*wSubAux,*wSlotBus,*wSlotDevice)

- **Parameter :**

  wBoardNo : [Input] Board number
  wBase : [Output] The base address of the board
  wIrq : [Output] The IRQ number that the board using.
  wSubVendor : [Output] Sub Vendor ID.
  wSubDevice : [Output] Sub Device ID.
  wSubAux : [Output] Sub Aux ID.
  wSlotBus : [Output] Slot Bus number.
  wSlotDevice : [Output] Slot Device ID.

- **Return:**

  Please refer to " Table A.1".

A-2-4 PIO_GetDriverVersion

- **Description :**

  This subroutine obtain the version number of PIODIO driver.

- **Syntax :**

  WORD PIO_GetDriverVersion(WORD *wDriverVersion)

- **Parameter :**

  wDriverVersion : [Output] Address of wDriverVersion

- **Return:**

  Please refer to " Table A.1".

A-2-5 ShowPIOPISO

- **Description :**

  This function will show a text string for this special Sub_ID. This text string is the same as that defined in PIO.H.

- **Syntax :**

  WORD ShowPIOPISO(wSubVendor, wSubDevice, wSubAux)

- **Parameter :**

  wSubVendor : [Input] SubVendor ID of the board
  wSubDevice : [Input] SubDevice ID of the board
  wSubAux : [Input] SubAux ID of the board.

- **Return:**

  Please refer to " Table A.1".